计算机系统结构实验指导书-LAB6

1. OVERVIEW

1.1 实验名称

简单的类 MIPS 多周期流水化处理器实现

1.2 实验目的

1．理解CPU Pipeline、流水线冒险(hazard)及相关性，在lab5基础上设计简单流水线CPU

2．在1. 的基础上设计支持Stall的流水线CPU。通过检测竞争并插入停顿（Stall）机制解决数据冒险/竞争、控制冒险和结构冒险

3．在2.的基础上，增加Forwarding 机制解决数据竞争，减少因数据竞争带来的流水线停顿延时，提高流水线处理器性能

PS：也允许考虑将Stall与Forwarding结合起来实现

4．在3. 的基础上，通过predict-not-taken 或延时转移策略解决控制冒险/竞争，减少控制竞争带来的流水线停顿延时，进一步提高处理器性能

PS：也允许考虑将2.、3.和4.合并起来设计

5． 在4.的基础上，将CPU 支持的指令数量从9条或16条扩充为31条，使处理器功能更加丰富（选做）

6．在5.的基础上，增加协处理器CP0，支持中断相关机制及中断指令（选做）

7．应用 Cache 原理，设计 Cache Line 并进行仿真验证（选做）

1.3 实验内容

本次实验将覆盖以下范围

1．CPU 流水化设计与软件实现与硬件实现

2．功能仿真，检验处理器的执行过程是否正确

3．上板验证（不做）

1.4 实验报告与验收办法

1） 认真完成实验报告，线上自查分析 CPU 指令执行的仿真结果并截图

2） 所有的实验报告和工程文件在第十一周星期二晚上 23 点前提交

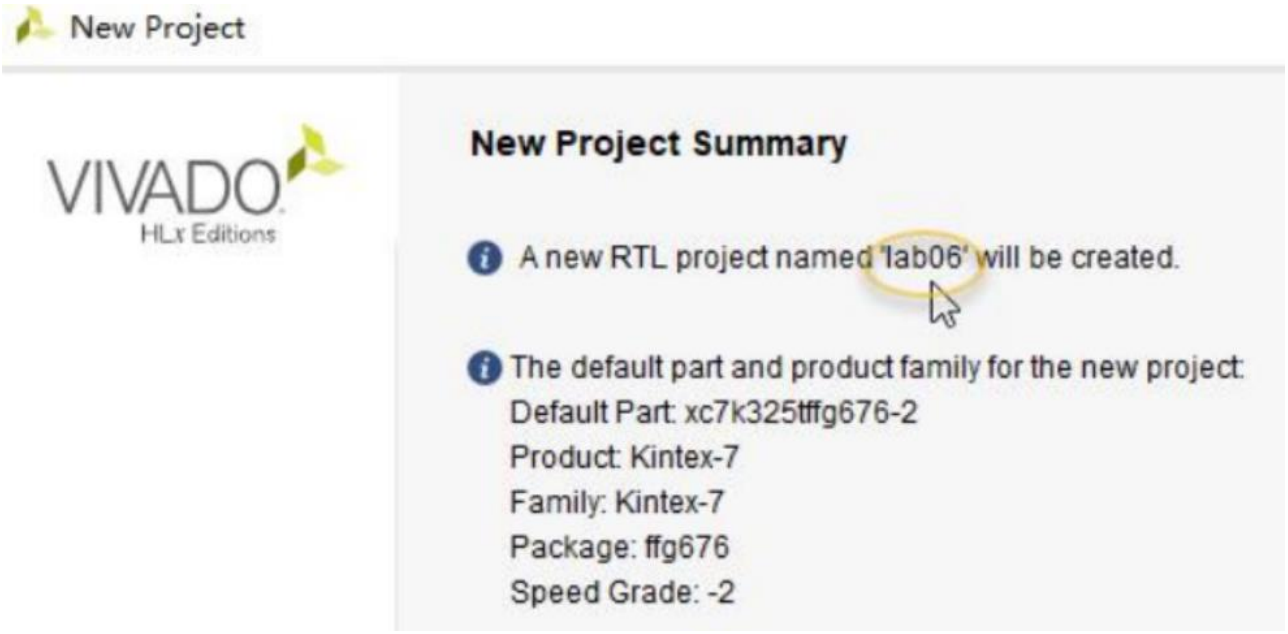
1.5 实验预计时间

480~640 分钟

2. 新建工程（简单流水线 CPU 软件实现的简单提示）

2.1 实验描述

2.1.1 新建工程



添加之前完成的功能模块（包括指令源文件）
后续可能新增自己所需的模块

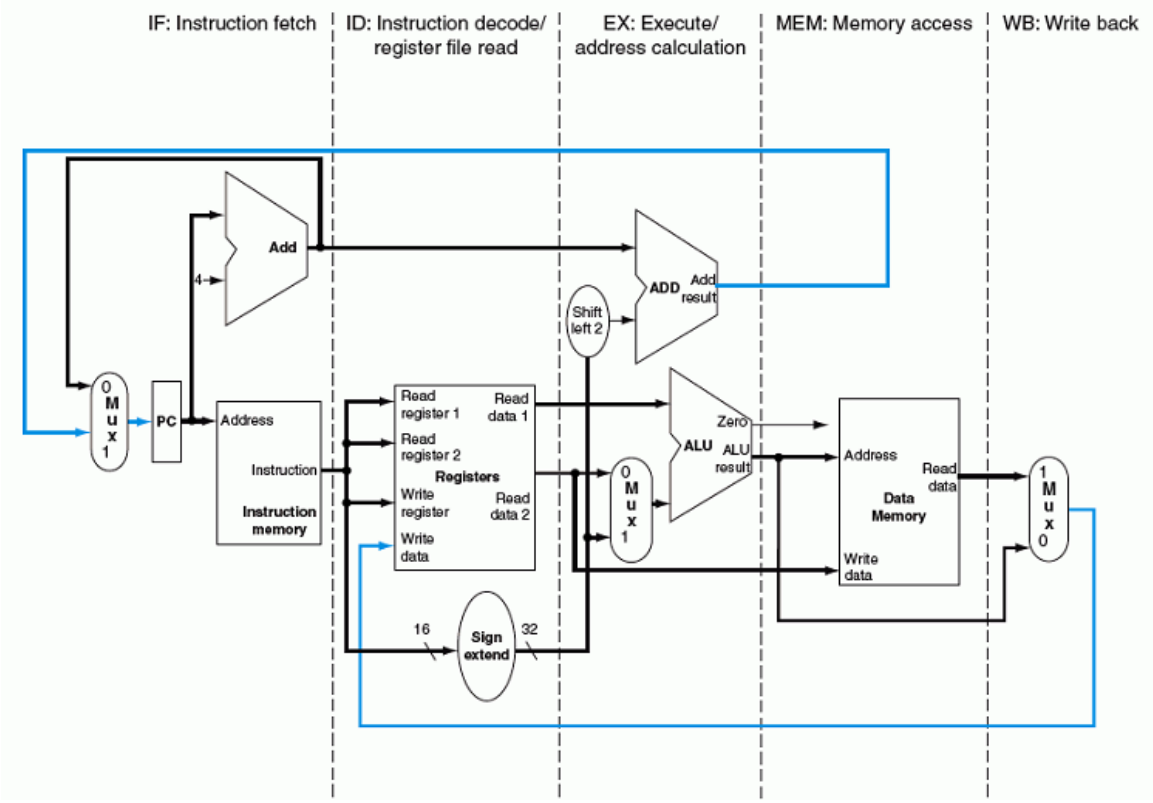
3. TOP 模块

3.1 实验描述

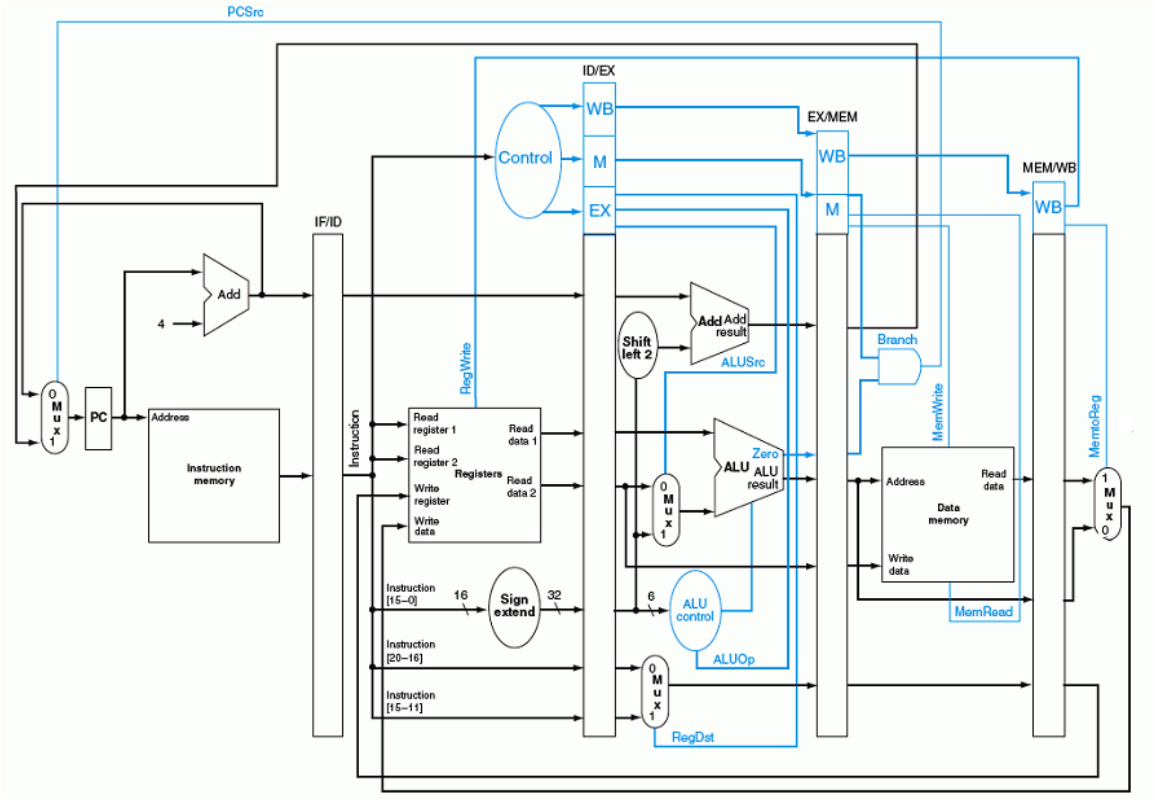
由于前几次实验已经完成了 CPU 各部分的主要功能，因此简单的流水化需要设计流水线的 Top 模块（包括修改 Control 模块，以及修改模块间互联的定义以及增加可能需要用到的子模块）。

3.1.1 模块描述

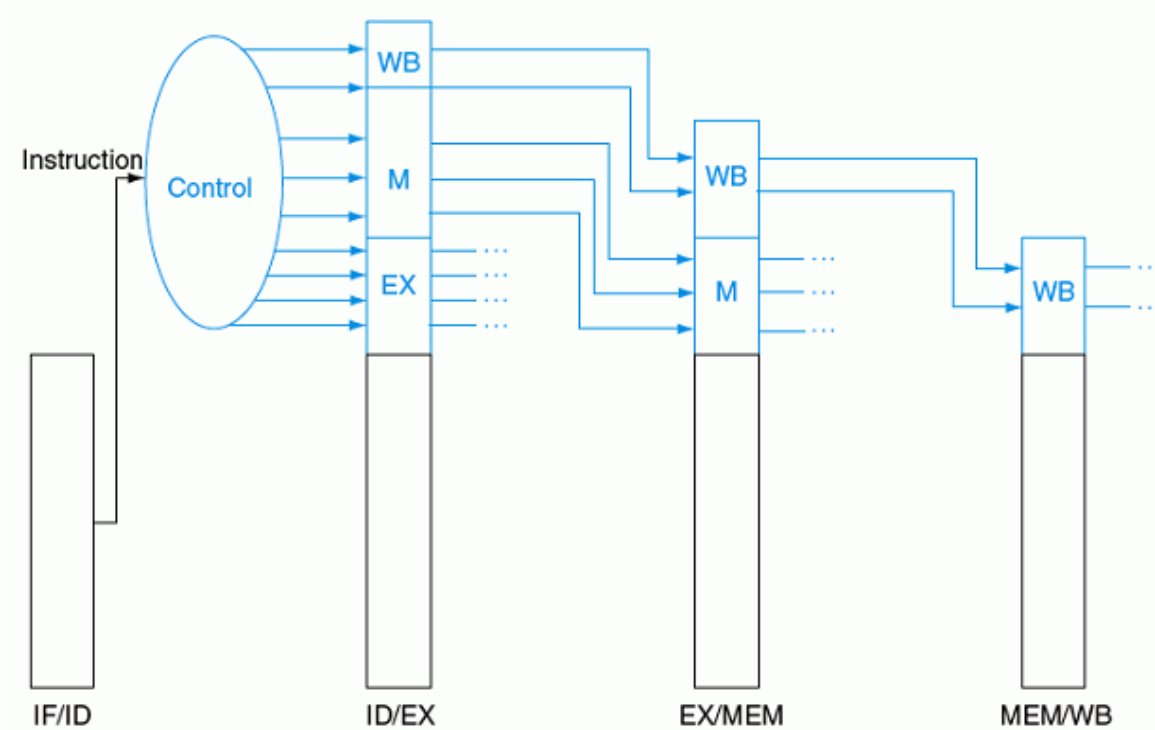
下面是流水的主要结构：



如图插入 4 个寄存器，将单周期 CPU 进行分割为 IF，ID，EX，M，WB 五大部分。这样将单周期的处理器运行分成 5 个周期完成，添加用于控制数据流的控制器，以实现 5 级流水线，包括 Fetch，Decode，Execute，Memory 和 Writeback



其中 Control 的输出需要被被加入流水线寄存器保存下来，以供后续每级流水使用。如下所示：



Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

3.1.2 修改与新增模块

修改模块

编写自己所需的模块

3.1.3 新建顶层模块

编写 Top 层模块，将各个模块互联起来

PS：1）和 Lab 5 相比，Top 模块中的主要变化处是什么

2）之前的模块是否要修改

3）另外，由于 MEM 级的 Branch 会影响 PCSrc 的值，从而影响下次 PC，因此需要为 Control 加入 RESET 功能，将 Branch 置零

4）由于各种变量名称极为复杂，推荐在着手编码之前为自己选择一套命名规范

5）在实现实验目的 2.，3.，4.的内容时建议把 1.的代码或工程备份一遍才开始

3.1.4 仿真测试

1. 编写汇编代码，推荐使用自己编写的的指令测试程序和数据。
   或者可以使用下面简单的参考指令程序。注意，需要去除中间的 Data Hazard：

```
lw $1, 40($0)    ; 1
lw $2, 44($0)    ; 5
lw $3, 48($0)    ; 8
add $4, $1, $2   ; $4=6
sub $5, $3, $1   ; $5=7
and $6, $2, $1   ; $6=1
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
or $7, $3, $1     ; $7=9
slt $8, $3, $1    ; $8=0
beq $0, $0, end ; to end
add $9, $7, $8   ; $9=9, not executed
end:
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
lw $10, 40($0)  ; 1
```

2. 将上述汇编代码转译为二进制机器码，保存为文件（存放到最后一级子文件夹 xsim 里），文件名自定，这里假定是 instruction.txt；

   将下列操作数据

   1
   5
   8

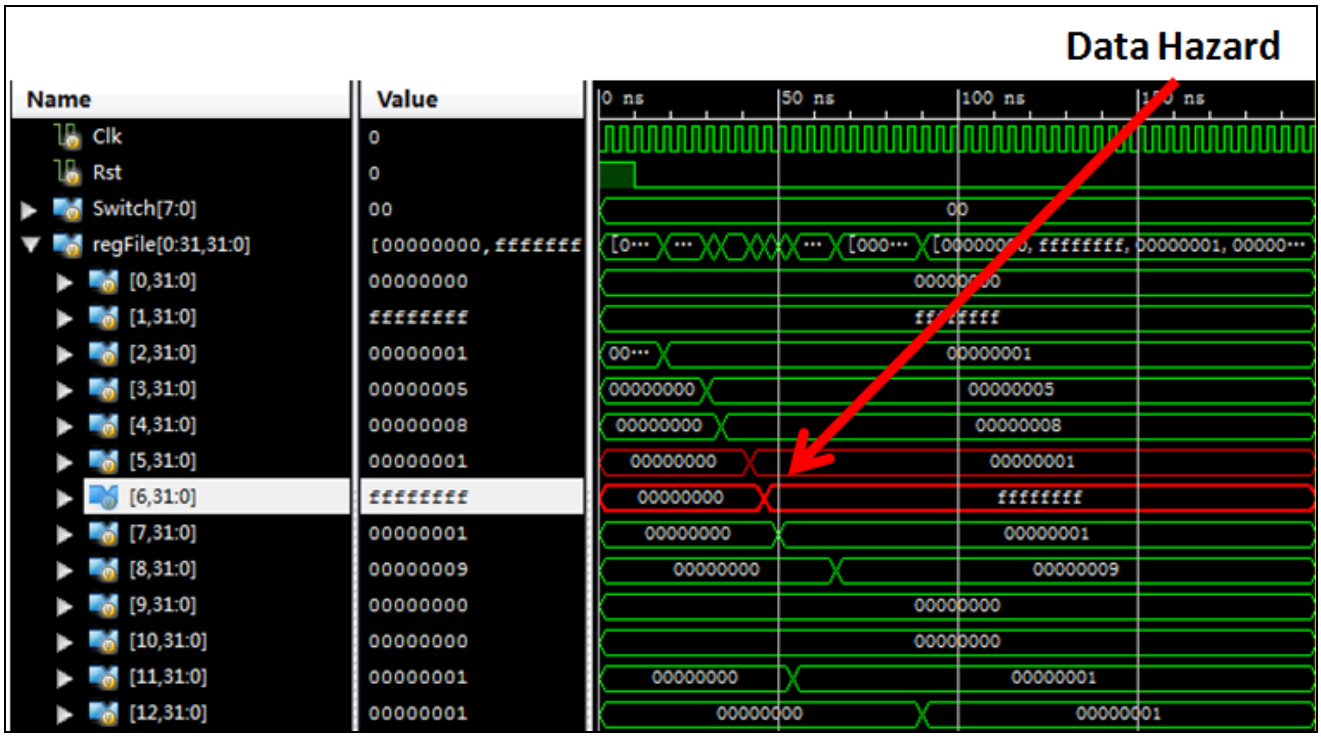
   假设保存为 data.txt

然后在顶层模块的激励源文件中（Top_tb）中加入下面初始化语句：

```
Initial begin
        $readmemb("instruction.txt", InstMemFile);
        $readmemb("data.txt", memFile);
end
```

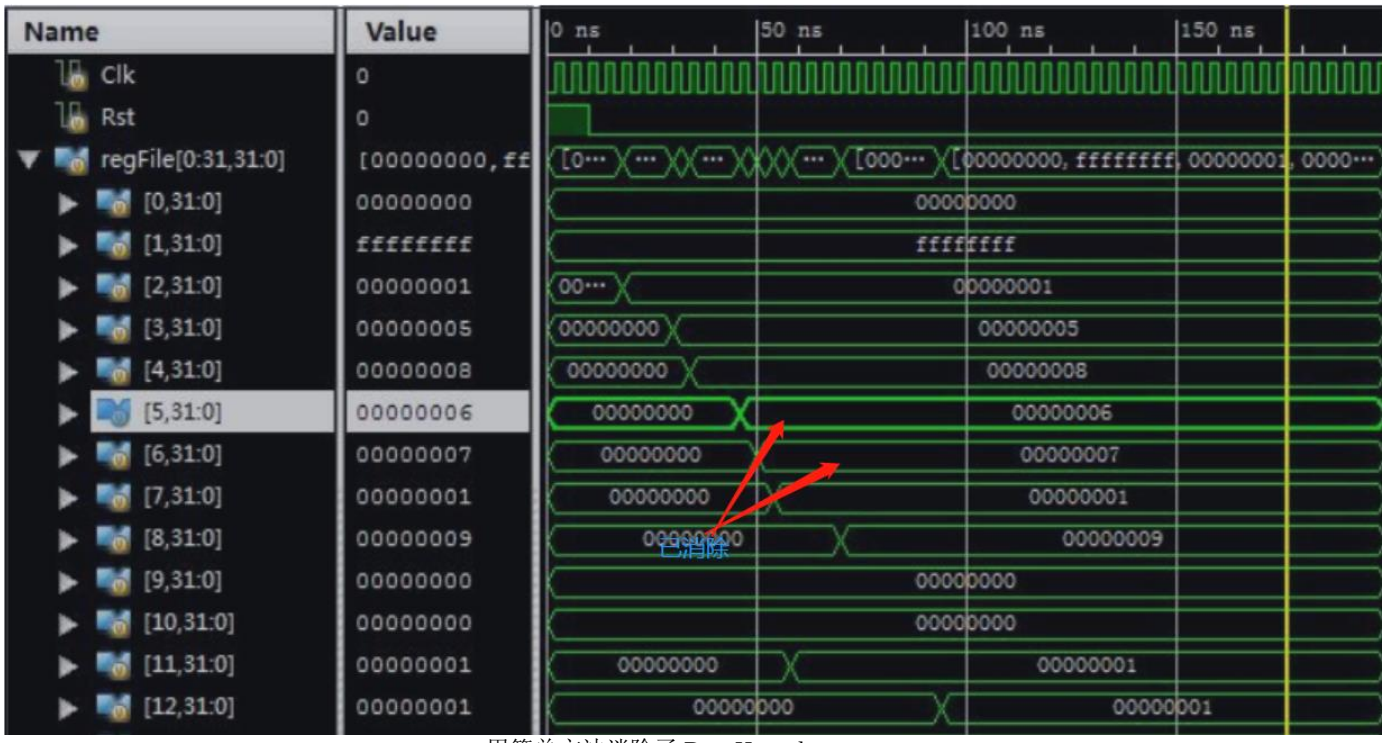
3. 编写 Top 的测试文件，添加时钟激励和其它输入信号并初始化

4. 下面给出基于上述指令运行的仿真参考样例：



出现了 Data Hazard

5. 修改指令后（软件实现 法），重新仿真，可看到数据竞争已消除：



用简单方法消除了 Data Hazard

3.2 实验报告

按指定时间提交